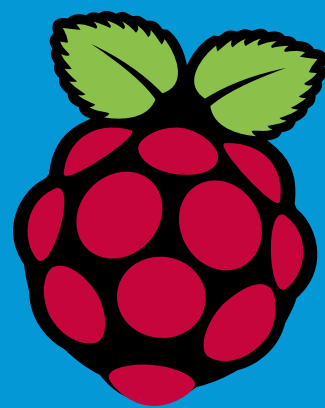
LA RIVISTA RASPBERRY PI UFFICIALE

La rivista ufficiale Raspberry Pi in italiano, da RaspberryItaly.com

Numero 58 Giugno 2017 www.raspberryitaly.com

GUIDA MAKER MINECRAFT

Incredibili idee per progetti di hack, codice e make

Gratuito!

The MagPi magazine is published by Raspberry Pi (Trading) Ltd., Mount Pleasant House, Cambridge, CB3 0RN. ISSN: 2051-9982

L'UNICA RIVISTA PI SCRITTA DALLA COMUNITÀ RASPBERRY PI

GUIDA MAKER MINECRAFT

Vuoi fare di più con Minecraft Pi?
Abbiamo alcuni progetti eccellenti da farti provare, sia per i novizi che per esperti!

Lo stretto Necessario

- Raspberry Pi (qualsiasi)
- Raspbian ultima versione
- Monitor
- Tastiera e mouse

Amiamo quanto sia facile modificare Minecraft su Raspberry Pi. Con delle librerie Python incorporate che consentono di modificare Il mondo in cui stai giocando, le possibilità sono quasi infinite!

Nel numero 41 abbiamo riportato di alcune modifiche di Minecraft, e in questo numero ritorniamo sull'Hacking di Minecraft con cinque nuovi progetti che fanno uso di differenti aspetti del gioco. Che si tratti di utilizzare una diverso metodo di programmazione, ad esempio EduBlocks, o usare schede RFID o una telecamera per legare Minecraft al mondo reale, lo abbiamo.

Accendi il tuo Raspberry Pi e Preparati a piegare Minecraft alla tua volontà.

MINECRAFT PI LE BASI

I tuoi primi passi nell'hacking di Minecraft Pi

La programmazione di Minecraft in Python, fa uso di speciali API, che ti consentono di controllare, alterare e interagire con il mondo Minecraft. Funzionano persino mentre usi il gioco e ti consentono di eseguire le seguenti operazioni:

- Prendere la posizione del giocatore
- Cambiare (o impostare) la posizione
- Identificare il tipo di blocco
- Cambiare un blocco
- Modificare l'angolo della telecamera
- Mandare messaggi al giocatore

Ciao Mondo!

La cosa più semplice che puoi fare è visualizzare un messaggio al giocatore (tu!) nel mondo di Minecraft. Ecco come lo puoi fare…

01. Accedi al menu di Minecraft premendo il tasto **ESC**, ma lasciando il gioco in azione.

02. Apri IDLE cliccando Menu > Programming > Python 3.

03. Usa File > New Window per creare un nuovo programma e poi salvalo come **hellominecraftworld.py**.

04. In testa al tuo programma, digita il codice seguente per importare il modulo 'minecraft', che ti consentirà di usare le API e comunicare col gioco:

```
import mcpi.minecraft as minecraft
```

05. Crea un collegamento tra il tuo programma e Minecraft e chiamalo `mc`:

```
mc = minecraft.Minecraft.create()
```

06. Usa il collegamento Minecraft e la funzione `postToChat()` per inserire un messaggio nella finestra di chat:

```
mc.postToChat("Ciao Mondo Minecraft")
```

07. Lancia il tuo programma cliccando su Run > Run Module o premendo **F5**.

Torna a Minecraft, vedrai il messaggio 'Ciao Mondo Minecraft' sullo schermo. Dovrai essere veloce, però, perché dura solo dieci secondi. Prova a scrivere altre parole e frasi a Minecraft.

Scrivi messaggi in Minecraft usando un codice molto semplice

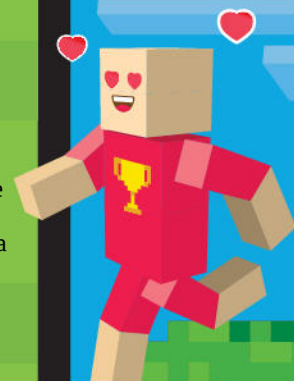
Blocchi e posizioni

Minecraft è un mondo di blocchi, tutti di circa 1 m × 1 m × 1 m. Il giocatore e ogni blocco nel mondo hanno una posizione fatta di x,y e z; x e z sono le posizioni sul piano orizzontale, e y è la verticale. Il giocatore parte dalla posizione x = 0, y = 0, z = 0, che è il punto di origine, e la posizione corrente del giocatore è mostrata In alto a sinistra sullo schermo.

Ogni blocco, in Minecraft, ha una posizione X, Y, Z

Aggiungi il codice seguente al tuo programma Ciao Mondo Minecraft per teleportare il giocatore (chiamato Steve) alla posizione x = 0, y = 50, z = 0, che lo piazzerà in aria a 50 blocchi di altezza:

```
mc.player.setPos(0, 50, 0)
```

Puoi utilizzare spezzoni di codice simili per cambiare, nello stesso modo, posizione a dei blocchi sulla mappa. Per cancellare un blocco, puoi modificarlo in un blocco di aria – è così che il vuoto viene inteso in Minecraft!

MARC SCOTT
Marc è a capo dello sviluppo Curriculum alla fondazione Raspberry Pi, gli piacciono anche i giochi pirotecnici.
raspberrypi.org

IL GRANDE PIANO MINECRAFT

Scatena il tuo Tom Hanks interiore su questo pianoforte gigante

In questo progetto creerai un pianoforte dentro Minecraft, che può essere suonato quando Steve (il giocatore) cammina sopra i tasti.

Questo progetto utilizza Sonic Pi per generare la musica, Minecraft per visualizzare il pianoforte, che funziona anche da input; infine utilizza Python per creare il pianoforte e consentire la comunicazione tra Sonic Pi e Minecraft.

PASSO 1

Ricevere messaggi in Sonic Pi

Il primo passo in questo progetto è provare ad inviare delle note da Python verso Sonic Pi. Questo è possibile perché Sonic Pi utilizza Open Sound Control (OSC). Questo metodo consente ai sintetizzatori digitali di comunicare tra loro attraverso una rete dati.

La prima cosa da fare è dire a Sonic Pi di ascoltare i messaggi in arrivo. Carica Sonic Pi, scegliendo Menu > Programming > Sonic Pi, poi clicca su Buffer 0 per iniziare a scrivere codice.

Avrai bisogno solo di poche linee di codice nel tuo file Sonic Pi – le trovi nel listato **MC_piano_sound** (a pag. 21). Questo dice a Sonic Pi di stare in ascolto delle note e di suonarle immediatamente. Puoi anche lanciare lo script adesso, ma non accadrà ancora nulla.

PASSO 2

Mandare messaggi a Sonic Pi

Crea un nuovo file Python 3 cliccando su Menu > Programming > Python 3 (IDLE), e successivamente cliccando sul menu File > New File. Hai bisogno del modulo python-osc per questo progetto, installalo col comando:

```
sudo pip3 install python-osc
```

Le prime due righe di **piano.py** (page 21) importano tutti i metodi necessari per il programma.

```
from pythonosc import osc_
message_builder
from pythonosc import udp_
client
```

Adesso dovrai creare un oggetto che invierà i messaggi. Open Sound Control consente

MINECRAFT CON : SONIC PI

Sonic Pi è un linguaggio di programmazione, già compreso in Raspbian, cheti permette di creare musica. Può anche collegarsi a Minecraft PI, cosa che lo rende perfetto per il modding.

ai computer di parlare tra loro, ma lo useremo per far parlare Python con Sonic Pi.

Poiché entrambi i programmi sono sullo stesso Raspberry Pi, puoi utilizzare l'indirizzo locale del Raspberry Pi per istruire Python su dove inviare il messaggio: è 127.0.0.1, e sarà sulla porta 4559. In **piano.py**, vedrai una riga come questa:

```
sender = udp_client.
SimpleUDPClient('127.0.0.1',
4559)
```

Questa invia il segnale al posto giusto. Quando viene attivata la funzione **play_note**, allora sa dove inviare la nota.

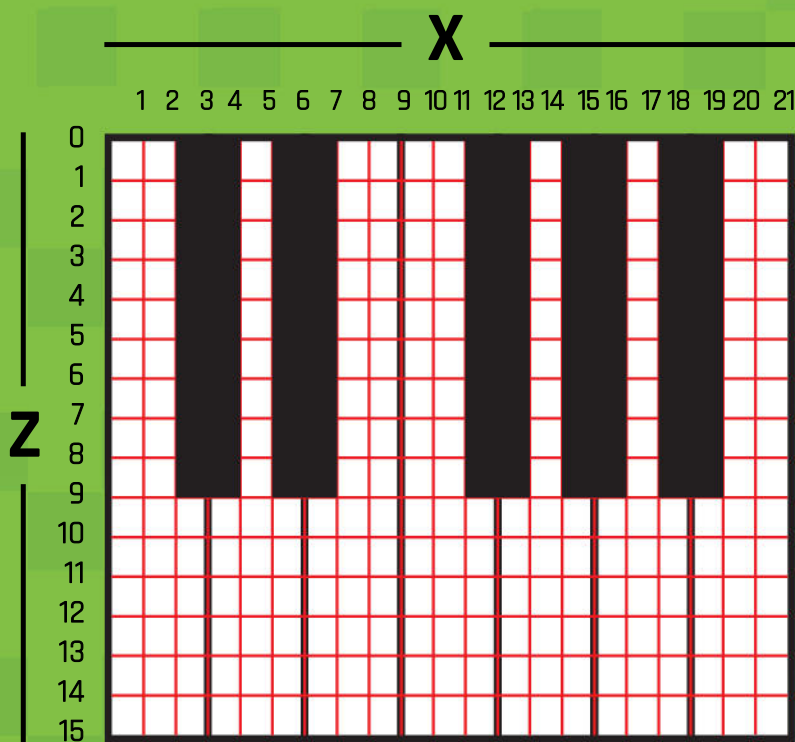
PASSO 3

Fare i tasti del piano

Costruire un piano in Minecraft può sembrare un compito un po' scoraggiante, quindi è più facile provare a semplificarlo in pezzi molto più piccoli. Questo è un Processo che gli esperti chiamano decomposizione.

Una tastiera di pianoforte comprende gruppi ripetuti di sette tasti bianchi e cinque neri – in sostanza le ottave. Realizzare ciascuno di questi elementi uno alla volta, ti permetterò di costruire facilmente la tastiera.

Il nostro codice funzionerà controllando la posizione del giocatore su tutti tre piani, in modo da poter assegnare la premuta di un tasto a specifiche coordinate. Questo viene fatto con la riga:

```
player_x, player_y, player_z
= mc.player.getTilePos()
```

PASSO 4

Pianificazione della tastiera

È sempre una buona idea fare uno schizzo veloce di quello che vuoi costruire prima di iniziare a posizionare blocchi nel mondo di Minecraft. Ecco lo schema (qui sopra) di un'ottava di una tastiera, che mostra le posizioni x e z dei blocchi.

PASSO 5

Fare un po' di spazio

A seconda di dove ti trovi nel mondo di Minecraft, potresti trovare che il tuo piano potrebbe essere stato creato al centro di una montagna. Per impedirlo, è possibile ripulire uno spazio con una funzione bulldozer che riempirà di aria un'area a forma di cubo, intorno al giocatore. La trovi in **piano.py**:

```
def bulldozer(x, y, z):
    mc.setBlocks(x - 30, y -
3, z - 30, x + 30, y + 20, z
+ 30, 0)
```

PASSO 6

Building black keys

Il codice utilizza una funzione chiamata **black_key** per realizzare i tasti neri del piano. La funzione avrà bisogno di sapere dove realizzare il tasto nero, avrà quindi bisogno di tre parametri. Questi parametri saranno la posizione x, y e z nel mondo Minecraft, dove il tasto deve essere realizzato.

Il passo successivo è quello di utilizzare la funzione **setBlocks**, per impostare alcuni blocchi Minecraft neri. Se guardi il tasto nero all'estrema sinistra, puoi notare che è largo due blocchi

e lungo nove blocchi. Quindi, se il Il primo blocco è posto a delle coordinate x e z, allora quello alla sua destra sarà posto a x + 1, e quelli che si trovano sotto di esso da z + 1 fino a z + 8. Tutti i blocchi Possono essere posizionati 1 blocco al di sotto della posizione del giocatore: y – 1.

L'ossidiana sembra un materiale ragionevole da cui costruire i blocchi. Ha un ID blocco di 49, quindi il codice **setBlocks** apparirà come:

```
mc.setBlocks(x, y - 1, z, x
+ 1, y - 1, z + 8, 49)
```

PASSO 7

Realizzare i tasti bianchi

Dai un'occhiata al primo tasto bianco nello schema (pagina 19). È largo tre blocchi e lungo 15. Stavolta, devi impostare blocchi da x fino a x + 2, e da z fino a z + 14. Chiameremo questa funzione **white_key** per farlo useremo il blocco "mattonella bianca", che ha un ID blocco di 44, 7. Il 44 è il blocco mattonella, e le 7 dice a Minecraft che deve essere bianco.

```
mc.setBlocks(x, y - 1, z, x
+ 2, y - 1, z + 14, 44, 7)
```

PASSO 8

Creare una ottava

Una ottava è composta da sette Note bianche e cinque note nere. Come nello schema (a pagina 19), i blocchi si estendono da x a x + 18. Il ciclo **for** serve per mettere un tasto bianco ogni tre blocchi sull'asse x, da 0 a 18.

Ora puoi iniziare a creare la tua funzione ottava, mettendo un tasto bianco in ogni posizione fornita da **i**. Cerca in **piano.py** questa funzione:

```
def make_octave(x, y, z):
    for i in range(0, 19, 3):
        white_key(player_x +
i, player_y, player_z)
```

I prossimi sono i tasti neri. Puoi Utilizzare lo stesso sistema per piazzarli. Consulta nuovamente lo schema. Questa volta, i tasti neri devono essere posizionati a partire da x = 2. All'interno della funzione **make_octave**, possiamo aggiungere un altro ciclo **for**.

> "Una ottava consiste in sette note bianche e cinque note nere"

Per ora, c'è solo un tasto di troppo. È stato inserito a x = 8, e devi fare in modo che questo tasto venga ignorato. Un poco di selezione condizionale può aiutare in questo caso. Se il valore di **i** è 8, allora la funzione **black_key** non deve essere richiamata. Un altro modo di dirlo è se **i** non è uguale a 8, La funzione **black_key** deve essere richiamata. Quindi aggiungiamo la condizionale **if i != 8:** alla funzione.

PASSO 9

Creare ancora una ottava

Ora mettiamo tutto assieme. Dopo tutte le funzione che abbiamo creato, possiamo ora richiamarle nel codice e utilizzare tre righe per Impostare tutto. Prima usare il buldozer sull'area, Poi realizzare il pianoforte, e poi sull'ultima riga impostare la posizione del giocatore, in modo che Steve venga messo al centro del pianoforte.

Ora, quando salvi e lanci il tuo codice, dovrebbe apparire un'ottava di pianoforte sotto i tuoi piedi. Ogni volta che tu lanci il codice, una nuova ottava sarà prodotta.

PASSO 10

Suona il tuo piano

Il prossimo passo è quello di ottenere che il pianoforte suoni una nota quando Steve cammina su di un tasto. Questo è gestito dal grande loop **while**. Parte controllando costantemente la posizione attuale di Steve.

Successivamente, trova il blocco sotto i piedi di Steve. Il Il problema è che i tasti bianchi Sono solo mezzo blocco di altezza.

Se Steve è in piedi su una mattonella bianca, a causa della loro piccola altezza, **block_below** termina, a causa dell'aria che è sotto il piano. Gestiamo questo aspetto con una condizionale, e controlliamo se il blocco sotto non è un tasto bianco o nero, che è quello che fa questo pezzetto di codice:

BIG

Perché è chiamato big piano? Beh, prima di tutto, è grande, e In secondo luogo è da una scena famosa di un vecchio film chiamato ***Big (da grande)*** dove due personaggi suonano musica su un grande piano in FAO Schwarz, un negozio di giocattoli di New York. Una versione del pianoforte Rimase lì fino a quando il negozio non è stato, recentemente, chiuso.

```
block_below =
mc.getBlock(new_x, new_y - 1,
new_z)
if block_below != 44 and
block_below != 49:
    block_below =
mc.getBlock(new_x, new_y,
new_z)
```

Poi, troviamo la posizione di Steve Rispetto alla posizione del piano. Il Il pianoforte è stato posizionato a **player_x**, ma ora Steve si trova a **new_x**.
Sottraendo una dall'altra, troverai dove si trova Steve sull'ottava del piano.

Compreso questo, si tratta solo di una lista di note da essere suonata. A partire da C centrale, le note bianche hanno i valori MIDI di 60, 62, 64, 65, 67, 68 e 71. Le note nere sono i valori MIDI mancanti tra le note bianche. Puoi megttere uno 0 in **black_notes**, in quanto ve ne sono solo cinque, sulla tastiera.

La nota bianca specifica da suonare, se Steve è in piedi sulla nota bianca, può essere trovata dividendo la sua posizione x relativa per -3 e poi ignorando le cifre decimali. Questo tipo di divisione si chiamasi chiama floor division, e in Python può essere fatta utilizzando l'operatore //, così:

```
if block_below == 44:
    notes_along = relative_
position // -3
    play_note(white_
notes[notes_along])
```

Per trovare la nota nera da suonare, sottraiamo 1 dalla posizione relativa di Steve, la dividiamo con la floor division per 3, e quindi sottraiamo di nuovo 1. Questo perché le note sono larghe solo due blocchi.

E questo è tutto. Prova a lanciare il codice e poi muoviti sui blocchi. Finché Sonic Pi è aperto e fa girare il tuo script iniziale, dovresti sentire il pianoforte suonare ogni volta che Steve passa su un particolare tasto.

Linguaggio
>PYTHON 3

NOME DEL FILE:
MC_piano_sound
piano.py

DOWNLOAD:
magpi.cc/MinecraftMaker

MC_piano_sound

```
set_sched_ahead_time! 0
live_loop :listen do
  message = sync "/play_this"
  note = message[:args][0]
  play note
end
```

piano.py

```
from pythonosc import osc_message_builder
from pythonosc import udp_client
from mcpi.minecraft import Minecraft
from time import sleep

sender = udp_client.SimpleUDPClient('127.0.0.1', 4559)
mc = Minecraft.create()

player_x, player_y, player_z  = mc.player.getTilePos()

def bulldozer(x, y, z):
  mc.setBlocks(x - 30, y - 3, z - 30, x + 30, y + 20, z + 30, 0)

def black_key(x, y, z):
  mc.setBlocks(x, y - 1, z, x + 1, y - 1, z + 8, 49)

def white_key(x, y, z):
  mc.setBlocks(x, y - 1, z, x + 2, y - 1, z + 14, 44, 7)

def make_octave(x, y, z):
  for i in range(0, 19, 3):
    white_key(player_x + i, player_y, player_z)
  for i in range(2, 18, 3):
    if i != 8:
      black_key(player_x + i, player_y, player_z)

def play_note(note):
  sender.send_message('/play_this', note)
  sleep(0.5)

bulldozer(player_x, player_y, player_z)
make_octave(player_x, player_y, player_z)
mc.player.setPos(player_x + 8, player_y + 3, player_z + 12)

while True:
  new_x, new_y, new_z = mc.player.getTilePos()
  block_below = mc.getBlock(new_x, new_y - 1, new_z)
  if block_below != 44 and block_below != 49:
    block_below = mc.getBlock(new_x, new_y, new_z)
  relative_position = player_x - new_x
  white_notes = [60, 62, 64, 65, 67, 69, 71]
  black_notes = [61, 63, 0,  66, 68, 70]
  if block_below == 44:
    notes_along = relative_position // -3
    play_note(white_notes[notes_along])
  if block_below == 49:
    notes_along = ((relative_position - 1) // -3) - 1
    play_note(black_notes[notes_along])
```

MARC SCOTT

Marc è a capo dello sviluppo Curriculum alla fondazione Raspberry Pi, gli piacciono anche i giochi pirotecnici.

MINECRAFT SELFIES

Sorridi! Sei inquadrato e nel mondo di Minecraft

Cosa Serve

> Raspberry Pi Camera Module magpi.cc/28ljlsz

In questo tutorial userai il modulo Pi Camera per scattarti un selfie e poi, con un po' di codice Python3, ricreerai l'immagine su di un gigantesco muro di blocchi Minecraft.

PASSO 1

Importare alcuni moduli

Per questo progetto devi importare alcuni moduli. La maggior parte di questi sono già pre-installati in Raspbian, ma devi comunque installare skimage, aprendo il terminale e digitando:

```
sudo apt-get install
python3-skimage
```

Apri Python 3 (IDLE) dal Menu. Crea un nuovo file cliccando su File > New File. Copia il codice del programma da listato **minecraft_selfie.py**, che si trova a pagina 25. Importa il modulo picamera per controllare la fotocamera e skimage per analizzare l'immagine. Salva il file come **minecraft_selfie.py**.

PASSO 2

Fare un selfie

La prima fase è molto semplice. Userai il modulo Pi Camera per scattarti un selfie.

Dopo aver importato i moduli necessari, dichiara l'oggetto `camera`, e imposta la risoluzione con le due linee seguenti:

```
camera = PiCamera()
camera.resolution = (80,60)
```

Puoi usare una risoluzione maggiore ma l'esecuzione del programma sarebbe più lunga, anche se si utilizza una Pi 3.

Il programma apre un'anteprima della fotocamera, aspetta un poco e poi scatta una foto che sarà salvata come **selfie.jpg**. Questa è la prima parte dello script.

PASSO 3

Mappare i colori sui blocchi

Scarica la mappa dei colori (**magpi.cc/2pQJaHS**) e salvala nella stessa directory del tuo script Python. L'immagine è

MINECRAFT CON : RASPBERRY PI CAMERA MODULE

Il modulo fotocamera ufficiale per il Raspberry Pi funziona molto bene con Python, ciò significa che possiamo usarlo in Minecraft

Fig 1

di soli 7 × 7 pixel, ma l'abbiamo ingrandita per mostrarti come appare: **Fig 1**.

Ogni pixel sulla mappa dei colori ha lo stesso colore medio di un blocco Minecraft. Per esempio il blocco in alto a sinistra è scuro.

Devi caricare sia la mappa dei colori che il selfie nel programma perché essi vengano rappresentati con una corrispondente lista di numeri.

> "Ogni pixel nella mappa dei colori ha il colore medio di un blocco Minecraft"

Qui è dove il modulo skimage diventa utile:

```
selfie_rgb = io.imread("selfie.
jpg")
map_rgb = io.imread("colour_
map.png")
```

Le variabili diventano array, i quali saranno simili a questo:

```
array([[[ 86,  74,  46],
        [ 93,  69,  49],
        [ 90,  87,  87],
        [ 99,  84,  65],
        [ 74,  73,  68],
        [108, 105,  95],
        [106,  95,  87]],
```

Questo è un estratto della rappresentazione dei colori nella mappa dei colori. La prima riga – 86, 74, 46 – rappresenta il primo pixel nella mappa dei colori. Viene creato da 3 numeri: il primo è la quantità di rosso, il secondo la quantità di verde, e il terzo di blu. In questo caso, il risultato è il colore marrone. Viene chiamato colore RGB.

Adesso hai i valori RGB dei pixel sul tuo selfie e il colore dei blocchi nella mappa dei colori, se riesci a trovare nella mappa il colore che assomiglia di più a quello sul selfie, saprai quale blocco scegliere e piazzare.

PASSO 4

Trovare il colore più prossimo

Qui è dove diventa un po' più complicato. Ogni colore è composto da tre numeri che ti permetteranno di tracciare la posizione dei colori su un grafico. Il colore R – 137, G – 164, B – 123 è stato rappresentato su un grafico 3D (**Fig 2**).

Ora tutti i colori della mappa dei colori possono essere rappresentati sullo stesso grafico usando punti più piccoli n modo da poter identificare il colore originale (**Fig 3**).

Sarebbe logico pensare che il punto più vicino nello spazio 3D a quello del colore originale sia quello del colore che ci assomiglia di più. Purtroppo però questo non è vero. I valori RGB sono utili per descrivere colori ma non per compararli. Dai una occhiata a **Fig 4** e **Fig 5**.

Fig 2

Fig 3

Fig 4

Fig 5

Fig 6

(0, 0): (2, 0),
(0, 1): (3, 0),
(0, 2): (4, 0),
(0, 3): (5, 0),
(0, 4): (7, 0),
(0, 5): (14, 0),
(0, 6): (15, 0),
(1, 0): (16, 0),
(1, 1): (17, 0),
(1, 2): (21, 0),
(1, 3): (22, 0),
(1, 4): (24, 0),
(1, 5): (35, 0),
(1, 6): (35, 1),
(2, 0): (35, 2),
(2, 1): (35, 3),
(2, 2): (35, 4),
(2, 3): (35, 5),
(2, 4): (35, 6),
(2, 5): (35, 7),
(2, 6): (35, 8),
(3, 0): (35, 9),
(3, 1): (35, 10),
(3, 2): (35, 11),
(3, 3): (35, 12),
(3, 4): (35, 13),
(3, 5): (35, 14),
(3, 6): (35, 15),
(4, 0): (41, 0),
(4, 1): (42, 0),
(4, 2): (43, 0),
(4, 3): (45, 0),
(4, 4): (46, 1),
(4, 5): (47, 0),
(4, 6): (48, 0),
(5, 0): (49, 0),
(5, 1): (54, 0),
(5, 2): (56, 0),
(5, 3): (57, 0),
(5, 4): (58, 0),
(5, 5): (60, 0),
(5, 6): (61, 0),
(6, 0): (73, 0),
(6, 1): (79, 0),
(6, 2): (80, 0),
(6, 3): (82, 0),
(6, 4): (89, 0),
(6, 5): (103, 0),
(6, 6): (246, 0)

Anche se i quadrati grigio scuro e grigio chiaro nella **Fig 4** a sembrano essere colori molto simili, la **Fig 5** mostra che, in realtà, si tratta di 173 unità distinte. Sia i punti grigio chiaro che quelli grigio scuro, sono entrambi molto vicini al rosso (150 unità) e molto vicini tra loro. Per questa ragione, il confrontare i valori RGB non è molto utile, visto che colori che sono vicini l'uno nell'altro nello spazio 3D possono visivamente sembrano essere molto diversi.

PASSO 5
Convertire in Lab colour space

A causa di questa disparità nel 3D Spazio, convertiamo i valori RGB In quello che è conosciuto come Lab colour space. In Lab color space, la distanza tra i colori nello spazio 3D è molto simile alla nostra percezione di ciò che potrebbe essere chiamato colori simili.

Il modulo skimage effettua la conversione in Lab colour space da RGB colour space, in modo semplice. Ti serviranno solo queste due linee aggiuntive:

```
selfie_lab = color.
rgb2lab(selfie_rgb)
map_lab = color.
rgb2lab(map_rgb)
```

PASSO 6
Mappare i blocchi

La parte successiva del codice, riguarda la mappatura dei pixel dalla mappa di colore dei blocchi di Minecraft effettivi. È stato usato un dizionario, per questo.

I blocchi Minecraft hanno due valori associati; per esempio, Lo sporco è 2, 0. Lo 0 viene usato perché esiste un solo tipo di blocco di sporcizia in Minecraft. La lana ne ha molti tipi, con diversi colori, infatti può variare da 35, 0 fino a 35, 15.

La parte difficile è già stata fatta qui, per te. Se dai uno sguardo alla tabella di (**Fig 6**),puoi trovarci i valori di pixel dalla mappa di colore, con i corrispondenti blocchi Minecraft.

PASSO 7
Partire con le API Minecraft

Ora è tempo di mettere i blocchi. In primo luogo troviamo la posizione del giocatore. Poi arriva la parte interessante. Stai per iterare tutti i colori del **selfie_lab**, prima di tutto. Per farlo, puoi chiedere aiuto alla funzione **enumerate**, che terrà traccia della tua posizione nel selfie:

```
for i, selfie_column in
enumerate(selfie_lab):
```

```
    for j, selfie_pixel in

        distance = 300
```

Queste tre linee “spazzoleranno” ogni pixel nel selfie e memorizzeranno ogni valore dei pixel come selfie_pixel. la distanza sarà impostata a 300, e le coordinate di ogni pixel saranno salvate come **i**, **j**.

Successivamente, dovrai iterare ogni pixel nella mappa dei colori nella stessa maniera:

```
for k, map_column in
enumerate(map_lab):
  for l, map_pixel in
enumerate(map_column):
```

Ora la distanza tra I colori dei pixel, può essere calcolata:

```
delta = color.deltaE_
ciede2000(selfie_pixel,map_
pixel)
```

Se il delta è inferiore alla distanza impostata precedentemente, la distanza viene ripristinata come delta. Il blocco può quindi essere esaminato dal dizionario di colori che hai Impostato prima:

```
if delta < distance:
  distance = delta
  block = colours[(k,l)]
```

Fig 7

Fig 8

Ora, al di fuori di quella parte del ciclo, puoi impostare il blocco appropriato. Sarà impostato relativamente alla posizione del giocatore, ma abbastanza alto nell'aria:

```
mc.setBlock(x-j, y-i+60, z+5,
block[0], block[1])
```

Prova adesso ad eseguire il codice completo e guarda cosa succede. Potresti aver bisogno di affinare il posizionamento dei blocchi e sii paziente se non lo ottieni immediatamente immediatamente. Dovrai ottenere qualcosa di simile alla **Fig 7**.

PASSO 8 (extra)

Un algoritmo migliore (ma più lento)

Puoi ottenere una rappresentazione più accurata utilizzando un diverso algoritmo per il calcolo del valore delta. Sarà più lento, ma potrebbe darti un risultato migliore (quindi sii molto paziente). Sostituisci la riga...

```
delta = color.deltaE_
cie76(selfie_pixel,map_pixel)
```

...con la riga seguente:

```
delta = color.deltaE_
ciede2000(selfie_pixel,map_
pixel)
```

Guarda il miglioramento in **Fig 8**.

MINECRAFT PHOTOBOOTH

Vuoi fare di più con le telecamere e Minecraft? Esiste un progetto leggermente diverso che potresti chiamare cabina fotografica di Minecraft. In esso, programmi Minecraft in modo che ogni volta che Steve entra in una cabina per foto nel mondo Minecraft, scatta una foto con la camera nella vita reale. Dagli una occhiata qui: magpi.cc/2pkDgLF

minecraft_selfie.py

Linguaggio
>PYTHON 3

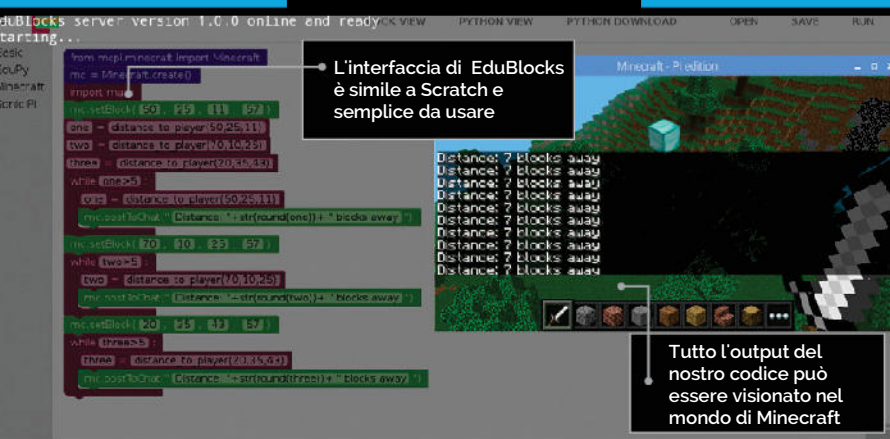
DOWNLOAD:
magpi.cc/MinecraftMaker

```
from picamera import PiCamera
from mcpi.minecraft import Minecraft
from time import sleep
from skimage import io, color

## Scattare una fotografia

camera = PiCamera()
camera.resolution = (80,60)
camera.start_preview()
sleep(15)
camera.capture('selfie.jpg')
camera.close()

## Rendering della immagine

### Carica il selfie e lo mappa
selfie_rgb = io.imread("selfie.jpg")
map_rgb = io.imread("colour_map.png")

### Converte a Lab

selfie_lab = color.rgb2lab(selfie_rgb)
map_lab = color.rgb2lab(map_rgb)

### Mappatura dei colori sulla mappa colori dei blocchi Minecraft
### La prima tupla sono le coordinate della mappa colori
### la seconda tupla sono i blocchi Mineccraft

colours={(0,0):(2,0),(0,1):(3,0),(0,2):(4,0),(0,3):(5,0),(0,4):(7,0),
(0,5):(14,0),(0,6):(15,0),(1,0):(16,0),(1,1):(17,0),(1,2):(21,0),(1,3):
(22,0),(1,4):(24,0),(1,5):(35,0),(1,6):(35,1),(2,0):(35,2),(2,1):
(35,3),(2,2):(35,4),(2,3):(35,5),(2,4):(35,6),(2,5):(35,7),(2,6):
(35,8),(3,0):(35,9),(3,1):(35,10),(3,2):(35,11),(3,3):(35,12),(3,4):
(35,13),(3,5):(35,14),(3,6):(35,15),(4,0):(41,0),(4,1):(42,0),(4,2):
(43,0),(4,3):(45,0),(4,4):(46,0),(4,5):(47,0),(4,6):(48,0),(5,0):
(49,0),(5,1):(54,0),(5,2):(56,0),(5,3):(57,0),(5,4):(58,0),(5,5):
(60,0),(5,6):(61,0),(6,0):(73,0),(6,1):(79,0),(6,2):(80,0),(6,3):
(82,0),(6,4):(89,0),(6,5):(103,0),(6,6):(246,0)}

## Itera immagine e poi la mappa. Trova il colore della mappa piu'
vicino poi cerca quel blocco e posizionalo

mc = Minecraft.create()
x, y, z = mc.player.getPos()

for i, selfie_column in enumerate(selfie_lab):
  for j, selfie_pixel in enumerate(selfie_column):
    distance = 300
    for k, map_column in enumerate(map_lab):
      for l, map_pixel in enumerate(map_column):
        delta = color.deltaE_cie76(selfie_pixel,map_pixel)
        if delta < distance:
          distance = delta
          block = colours[(k,l)]
    mc.setBlock(x-j, y-i+60, z+5, block[0], block[1])
```

JOSHUA LOWE

Josh è un 13enne nel Nord Ovest dell'Inghilterra a cui piace lo sviluppo Software e progettare prodotti per aiutare gli altri.
edublocks.org / @all_about_code

PROGRAMMARE MINECRAFT CON EDUBLOCKS

Crea la tua caccia al tesoro in Minecraft con il nuovo programma EduBlocks - un modo semplice per passare da Scratch a Python

L'interfaccia di EduBlocks è simile a Scratch e semplice da usare

Tutto l'output del nostro codice può essere visionato nel mondo di Minecraft

J osh ha creato EduBlocks pensando a un modo per dare la possibilità a tutti i bambini di scrivere codice Python tramite un semplice editor a blocchi simile a Scratch. L'obiettivo del progetto è quello di rendere la transizione da Scratch a Python è più facile per studenti e insegnanti, visto che attualmente non esiste una soluzione in grado di colmare questo divario.

EduBlocks è nato 15 mesi fa, come un sistema per aiutare gli insegnanti a fare di più in classe e aiutare i bambini a esplorare il mondo del Raspberry Pi tramite una interfaccia facile da usare. EduBlocks è un successo grazie all'aiuto della meravigliosa Comunità Raspberry Pi, che ha fornito spunti e risorse per il progetto.

In questo tutorial creeremo un gioco in cui tre diamanti sono nascosti in giro per il mondo Minecraft. Pensi che tu possa trovarli tutti e tre?

Iniziamo il progetto installando EduBlocks sul nostro Raspberry Pi.

Per farlo, dovrai aprire un terminale sul tuo Raspberry Pi e digitare il seguente comando:

```
curl -sSL get.edublocks.org
| bash
```

Per iniziare la tua avventura con EduBlocks, abbiamo bisogno di avviare l'applicazione. C'è un comodo link sul desktop, su cui puoi fare un doppio clic per avviare EduBlocks. C'è anche un link nel menu di programmazione che può essere utilizzato per avviare l'applicazione.

MINECRAFT CON : EDUBLOCKS

EduBlocks è il sistema per colmare il divario tra Scratch e Python, utilizzando blocchi di codice preimpostati

Blocchi usati per scrivere in Python

EduBlocks impiegherà un po' di tempo per caricarsi, ma una volta fatto, ti verrà presentata la interfaccia utente di EduBlocks, che consiste in una grande area di lavoro. È qui che metteremo il nostro codice per sviluppare il gioco. I blocchi di codice utilizzati per costruire il nostro progetto si trovano sul lato sinistro dello schermo: basta trascinare e rilasciare i blocchi sull'area di lavoro per creare la sequenza di codice per il nostro gioco. Dobbiamo lavorare nella block view, ma in qualsiasi momento possiamo passare alla visuale Python in modo da poter vedere il codice Python, creato utilizzando i blocchi.

MEHDI IMANI MASOULEH

Ingegnere elettronico e uno dei Fondatori originali di Piper, un computer portatile basato su Raspberry Pi che Insegna l'elettronica attraverso Minecraft! buildpiper.com

CAMBIA SKIN MINECRAFT CON LE CARTE RFID

Questo progetto consente di modificare il protagonista di Minecraft Pi passando diverse carte RFID su di un lettore

MINECRAFT CON: RFID

la radio-frequencey identification, o RFID, permette ai computer di riconoscere un codice tramite onde radio. Viene usato nella tecnologia contactless

Cosa Serve

- 7 cavallotti maschio - femmina
- Mini breadboard
- modulo RFID - RC522
- Qualche tessera Mifare
- 8 pin angolati

Minecraft sul Raspberry Pi è preimpostato con il personaggio principale pettinato come Herobrine. In questo tutorial spiegheremo come hackerare Minecraft Pi per consentire la modifica della skin nel tuo personaggio preferito, utilizzando le schede RFID. Dopo aver costruito un semplice circuito e fatto qualche lavoretto software, potrai quindi toccare il lettore con la tua scheda RFID personalizzata, per poter cambiare immediatamente Il personaggio di Minecraft.

La skin di Minecraft è cambiata in quella mostrata sulla carta RFID!

La scheda RFID viene avvicinata al lettore RFID per aggiornare Il protagonista di Minecraft

PASSO 1
Realizzare il circuito

Cominciamo collegando il Modulo lettore RFID-RC522 al Raspberry Pi. Il connettore a otto pin deve essere saldato al lettore RFID se non viene fornito già pre-saldato. Inserisci l'estremità femmina del cavallotto jumper maschio-femmina sui piedini del GPIO di Raspberry Pi, mentre i maschi vanno sulla breadboard, come mostrato nello schema.

PASSO 2
Impostare l'RFID sul Pi

Per facilitare la comunicazione tra il modulo RFID e il Pi, dobbiamo prima configurare Il nostro Raspberry Pi in modo che la Peripheral Interface (SPI) sia abilitata. È necessario installare la libreria Python SPI hardware. Tutti questi passaggi sono spiegati in questa guida online: **magpi.cc/28LleQN**. È necessaria anche una libreria RFID per rendere più facile la comunicazione con il modulo RFID, ma questa è inclusa nella cartella del progetto.

PASSO 3
Scaricare le skin

Nella cartella del progetto è incluso un piccolo set di skin. Altre skin si possono trovare su **minecraftskins.net**. Scarica il tuo personaggio preferito sul tuo Raspberry Pi. Ricorda dove sono così li potrai spostare nella cartella corretta, nel passo 5.

PASSO 4
Fai la tua carta personaggio

Puoi disegnare il personaggio di Minecraft che hai scelto su di un foglio di carta oppure scaricare le immagini dei personaggi da **minecraftskins.net** e poi stamparle. Ritaglia il tuo personaggio e usa la colla in stick per incollarlo sulla card Mifare. Ripeti per ogni personaggio che desideri.

PASSO 5
Prendi il codice

Digitare il seguente comando per installare l'applicazione xdotool, che consente di gestire i task delle delle finestre, usando i comandi della shell:

```
sudo apt-get install xdotool
```

Verrà utilizzato nel codice per aggiornare la finestra di Minecraft, in modo che la skin venga aggiornata automaticamente. Scarica il progetto da GitHub (**magpi.cc/2pgdXio**) e piazzalo in una nuova cartella di progetto. Sposta le skin che hai salvato in precedenza nella cartella **skins** del progetto. lancia il file Python **Read.py** digitando **`sudo python Read.py`**, e passa ogni card Mifare sul lettore RFID. Questo dovrebbe testare il buon funzionamento del del circuito e darti l'ID univoco delle carte. Sostituisci gli ID delle carte nel codice con gli ID che letti eseguendo **Read.py**. Sostituisci la variabile **`skinFile`** in **charMinecraft.py** con il corrispondente nome del file skin nella cartella **skins**.

PASSO 6
Pronto a giocare!

Lancia Minecraft e avvia il codice Python aggiornato, digitando **`sudo python charMinecraft.py`** nel terminale. Assicurati che il gioco sia in visuale in terza persona, premendo il tasto **ESC** nel gioco e cliccando poi sul bottone third-person view. Appoggia la card Mifare sul lettore RFID per cambiare il personaggio. Una volta posizionata la carta, un messaggio sullo schermo dovrebbe informarti verso quale personaggio stai cambiando. la skin viene poi aggiornata e puoi continuare a giocare il tuo gioco preferito, con il tuo personaggio preferito!

charMinecraft.py

Linguaggio
>PYTHON

NOME DEL FILE:
charMinecraft.py

DOWNLOAD:
magpi.cc/2pgdXio

```
import RPi.GPIO as GPIO
import MFRC522
import signal
import mcpi.minecraft as minecraft
import time,os

# Rileva SIGINT per ripulire quando lo script viene abortito
def end_read(signal,frame):
  global continue_reading
  print "rilevato Ctrl+C, fine lettura."
  continue_reading = False
  GPIO.cleanup()

#Rimpiazza qui il nome del file della skin
skinFile=['ironman','default','batman','pig']
skinNames=['Iron Man', 'Herobrine', 'Batman']
idx=1;
winSizeX=1800 #imposta le dimensioni della finestra minecraft
winSizeY=800

#crea la connessione minecraft
mc = minecraft.Minecraft.create()

continue_reading = True
#Rimpiazza qui le card ID
UIDs=['160,41,83,122','144,24,1,118','176,221,21,124']

# Aggancia il SIGINT
signal.signal(signal.SIGINT, end_read)

# Crea un oggetto per la classe MFRC522
MIFAREReader = MFRC522.MFRC522()
i=0;

print "Premi Ctrl-C per fermare."

# Questo ciclo continua a controllare le card.
while continue_reading:

  # Scansiona per le card
  (status,TagType) = MIFAREReader.MFRC522_Request(MIFAREReader.PICC_REQIDL)

  # Se una card viene rilevata
  if status == MIFAREReader.MI_OK:
    print "Card rilevata"

  # Prendi l' UID della card
  (status,uid) = MIFAREReader.MFRC522_Anticoll()

  # Se abbiamo l'UID, continua
  if status == MIFAREReader.MI_OK:
    # Stampa l' UID
    print "Card UID: "+str(uid[0])+","+str(uid[1])+","+\
    str(uid[2])+","+str(uid[3])
    uid_str=str(uid[0])+","+str(uid[1])+","+str(uid[2])+","+str(uid[3]);

  try:
    idx=UIDs.index(uid_str)
    os.system('cp skins/'+ skinFile[idx]
      +'.png //home/pi/mcpi/data/images/mob/char.png')
    mc.postToChat('Skin changed to: '+skinNames[idx]+'!')
    i=i+1; #fai il refresh della finestra
    os.system("xdotool search --name 'Minecraft - PI'  windowsize "
      + str(winSizeX)+ ' ' +str(winSizeY+i%2))
    except ValueError:
      print("Oops! Non è nella lista!")
```

RICHARD JARVIS

Insegnante di computer science con la passione di fare e rompere le cose. Autore di appJar.
appJar.info

REALIZZA UNA GUI PYTHON PER MINECRAFT

Crea, usando appjar, una semplice GUI Python per controllare il tuo Minecraft

Cosa Serve

- appJar appJar.info

Noi tutti sappiamo quanto sia facile controllare Minecraft da Python, ma lo sapevi che è altrettanto facile creare una GUI (Interfaccia Grafica Utente) per fare la stessa cosa? Con appJar, e alcune righe di codice, puoi creare una semplice GUI per controllare il tuo mondo Minecraft.

PASSO 1
Impostazioni

Prima di tutto, devi installare appJar. Apri un terminale e digita **`sudo pip3 install appJar`**. Ora sei pronto a cominciare – apri l' IDLE (per Python 3), e cominciamo a programmare.

L'ordine, nella programmazione, è importante: prima importeremo le librerie Minecraft e appJar, poi uniremo insieme tutte le nostre funzioni. Infine, scriveremo Il codice per la nostra GUI.
Programmare una GUI è un processo sempre suddiviso in tre parti: creare la GUI, aggiungere e configurare i Widget, quindi avviare la GUI. Dopo di essa, non mettere codice, se non vuoi che venga eseguito dopo la chiusura della GUI.

PASSO 2
Chattiamo

Per iniziare, aggiungeremo una casella di testo e un bottone, per inviare un messaggio di chat. AppJar mette i widget in una griglia, quindi mettiamo entrambi i widget in fila 0, colonne 0 e 1.
Quindi, avremo bisogno di una funzione da chiamare quando il bottone viene premuto – prenderà il testo dalla casella e lo manderà a Minecraft. Il secondo parametro, quando si aggiunge un bottone, è il nome della funzione da richiamare quando viene premuto.

PASSO 3
Muoversi

Successivamente, aggiungeremo alcuni bottoni di movimento. Cambieranno semplicemente la nostra coordinata x, y, o z.
Andiamo a raggruppare i pulsanti in un **`LabelFrame`**, lo metteremo quindi nella prossima fila, e gli diremo di abbracciare entrambe le colonne. Poi posizioniamo i bottoni all'interno del **`LabelFrame`** – esso ha una propria griglia, quindi possiamo posizionare I bottoni proprio come prima.
Collegheremo tutti questi pulsanti a una nuova funzione. Il nome del Il pulsante viene passato come parametro alla funzione, quindi possiamo usare un operatore **`if`** per processare quale bottone è stato premuto e cambiare la giusta coordinata.

PASSO 4
Aggiornamenti di stato?

A tutti piace aggiornare il proprio stato, e appjar lo rende davvero semplice. Stiamo per creare un Barra di stato, e farle mostrare la nostra posizione attuale.
Vogliamo mantenere aggiornata la barra di stato mentre giochiamo, ma non possiamo utilizzare un ciclo, in quanto bloccherà la GUI

MINECRAFT CON: APPJAR

La libreria di appJar è un metodo per creare una interfaccia grafica utilizzando del codice Python. È molto facile da usare, come puoi vedere dal nostro script.

e il suo funzionamento. Invece, creeremo una funzione per aggiornare la barra di stato, e poi chiederemo a appjar di metterla in un loop per noi.

PASSO 5

Caduta blocchi

Vogliamo realizzare blocchi cadenti nel modo più semplice possibile, così metteremo un box opzioni per scegliere il blocco Da, e un bottone fallo cadere. La funzione collegata a questo bottone, semplicemente verificherà quale blocco è stato selezionato, trova il suo ID, E dice a Minecraft di posizionare tale blocco accanto al nostro personaggio.

Dovremo creare un dizionario dei nostri blocchi preferiti. La chiave sarà un nome facile da ricordare, e il valore sarà il corrispondente ID.

PASSO 6

Qualcosa dal menu?

Infine, aggiungiamo un po' di opzioni di menu, per avere lo stesso feeling di una vera e propria applicazione...

Avremo un menu per la creazione e il ripristino dei checkpoint, poi faremo un altro menu per cambiare l'angolazione della telecamera.

Funzionano proprio come i bottoni. Collegheremo tutti i menu alla stessa funzione, controlleremo il parametro per vedere quale menu è stato cliccato, quindi faremo la specifica azione. Aggiungeremo anche un paio di finestre di dialogo, per dare qualche riscontro delle scelte del menu.

code_5.py

Linguaggio
>PYTHON

NOME DEL FILE: code_5.py

DOWNLOAD: magpi.cc/MinecraftMaker

```
001. # importa le librerie
002. from appJar import gui
003. from mcpi.minecraft import Minecraft
004.
005. # connessione a Minecraft
006. mc = Minecraft.create()
007.
008. # 1 – FUNZIONE CHAT
009. def sendChat(btn):
010.     msg = app.getEntry("Chat")
011.     mc.postToChat(msg)
012.
013. # 2 – FUNZIONE MOVIMENTO
014. def move(btn):
015.   x, y, z = mc.player.getPos()
016.   if btn == "LEFT":
017.     x -= 1
018.   elif btn == "RIGHT":
019.     x += 1
020.   elif btn == "FORWARD":
021.     z -= 1
022.   elif btn == "BACKWARD":
023.     z += 1
024.   elif btn == "JUMP":
025.     y += 1
026.     z -= 1
027.
028.   mc.player.setPos(x, y, z)
029.
030. # 3 – FUNZIONE DI STATO
031. def updateStatus():
032.   x, y, z = mc.player.getPos()
033.   app.setStatusbar("X: " + str(x), field=0)
034.   app.setStatusbar("Y: " + str(y), field=1)
035.   app.setStatusbar("Z: " + str(z), field=2)
036.
037. # 4 - BLOCKS FUNCTION
038. BLOCKS = {"Stone": 1, "TNT": 46, "Torch": 50,
     "Diamond": 57}
039. def drop(btn):
040.   x, y, z = mc.player.getPos()
041.   z = z - 1
042.   height = mc.getHeight(x,z)
043.
044.   playerBlock = app.getOptionBox("Block")
045.   blockId = BLOCKS[playerBlock]
046.
047.   mc.setBlock(x, height, z, blockId)
048.
049. # 5 - MENU FUNCTION
050. def clickMenu(choice):
051.   if choice == "Create":
052.     mc.saveCheckpoint()
053.     app.infoBox("Save",
     "Checkpoint saved.")
054.   elif choice == "Restore":
055.     if app.yesNoBox("Restore",
       "Are you sure?"):
056.       mc.restoreCheckpoint()
057.   elif choice == "Normal":
058.     mc.camera.setNormal()
059.   elif choice == "Fixed":
060.     mc.camera.setFixed()
061.   elif choice == "Follow":
062.     mc.camera.setFollow()
063.
064. # create the GUI - must come first
065. app = gui("appJar Minecraft")
066. app.setLocation(100,100)
067.
068. # 1 - CHAT WIDGETS
069. app.addLabelEntry("Chat", row=0, column=0)
070. app.addButton("Send", sendChat, row=0, column=1)
071.
072. # 2 - MOVEMENT WIDGETS
073. app.startLabelFrame("Movement", row=1, column=0,
     colspan=2)
074. app.setSticky("NESW") # make buttons stick to
     all sides
075. app.addButton("FORWARD", move, row=0, column=1)
076. app.addButton("LEFT", move, row=1, column=0)
077. app.addButton("JUMP", move, row=1, column=1)
078. app.addButton("RIGHT", move, row=1, column=2)
079. app.addButton("BACKWARD", move, row=2, column=1)
080. app.stopLabelFrame()
081.
082. # 3 – WIDGET STATO
083. app.addStatusbar(fields=3)
084. app.registerEvent(updateStatus) # call
085. updateStatus in a loop
086.
087. # 4 – WIDGET BLOCCHI
088. app.addLabelOptionBox("Block", list(BLOCKS),
     row=2, column=0)
089. app.addButton("Drop", drop, row=2, column=1)
090.
091. # 5 – WIDGET MENU
092. app.addMenuList("Checkpoint", ["Create",
     "Restore"], clickMenu)
093. app.addMenuList("Camera", ["Normal", "Fixed",
     "Follow"], clickMenu)
094.
095. # lancia la GUI - must come last
096. app.go()
```